Assignment 2

Shahar Weinstock 300531712
Michal Ben Neria 038178950

The features

The list of features extracted in part 2:

All below features were extracted for each participant from the relevant source file. Most of the features were extracted as "average per day" in order to minimize the effect of some participants being in the experiment longer than others.

1. Average number of calls per day
2. Average number of SMSs per day
3. Average number of meetings per day, where a meeting is defined in our work as a transaction in the BluetoothProximity.csv file.
4. Average of distinct number of call contacts per day
5. Average of distinct number of SMS contacts per day
6. Average of distinct people a user meets per day
7. Proportion of incoming calls
8. Proportion of incoming SMSs
9. Average duration of calls- before extracting this feature all duration missing values were assigned to the general averaged duration of all participants.

KNN is very sensitive to scaling issues, thus we performed a min-max ([0,1]) scaling upon all features.

The list of features extracted in part 6:

We built a different graph from each of the source files. The graphs consisted only recognized participants as nodes. The edges represent the existence of communication (calls/SMSs/meetings) of at least 5 days (configured using the variable min-num-of-days). This was done for filtering the possible outliers, for example- two participants that talked 20 times but only in one day, such that an average weight is very high although this connection is not that strong. The below features were extracted from the corresponding graphs for each participant (node).

1. Call's PageRank
2. SMS's PageRank
3. Meeting's PageRank

*Note that the PageRank scores are referred to the in/out degree only and not considering a weight derived from the edges' attribute (it turns out that GraphLab still don't support such functionality).

4. In degree of average calls per day
5. Out degree of average calls per day
6. In degree of average SMSs per day
7. Out degree of average SMSs per day
8. Degree of average meetings per day- no in/out separation made here because the meetings graph is undirected.

Here, again, a min-max scaling applied to all features. A few outliers were removed before scaling, using manual examination.

At this point, we had too many features after the combination made in part 7. Hence, to overcome an over-fitting issue, we had to perform a feature selection step before applying the ML algorithm (part 8). The feature selection made by using 5-folds cross-validation of Logistic Regression- each feature which its weight was very close to zero in most of the folds was eliminated. Eventually, we used only the following 7 features in part 8:

1. Call's PageRank
2. Meeting's PageRank
3. In degree of average calls per day
4. Out degree of average calls per day
5. In degree of average SMSs per day
6. Out degree of average SMSs per day
7. Degree of average meetings per day

The implemented algorithm

The algorithm we chose to implement in part 3 is weighted KNN.

We applied it to the features extracted during part 2.

We implemented the distributed algorithm using Pyspark as follows:

Weighted KNN input:

The RDD of the training set,

The predicted line- the new line we wish to predict its label,

The value of K nearest neighbors

We start with a MapReduce step for finding the K nearest neighbors:

Mapper input: RDD record

Mapper output: (key: 1, value: a list with a tuple of the Euclidean distance, between the predicted line and the RDD record, and the RDD record's label)
All mappers outputs will be handled by the same reducer because of the Map key output we produced (1).
Reducer (by key) input: (v1, v2), where each one of them is a list of tuple(s) of the Euclidean distance, between the predicted line and the RDD record(s), and the RDD record's label.
Reducer (by key) output: concatenate v1 and v2 and sort the new list. Then:

If the new list length is greater than K:

Return a list of the first K (sorted) tuples of the Euclidean distance and the label

Else:

Return the new (concatenated) list

After the reduce step is finished, we have the K nearest neighbors' Euclidean distance and label. We then convert the distance values to similarity weights and perform a weighted calculation to predict the relevant label.
Weighted KNN output: the label of the predicted line.

The accuracy of the implemented algorithm with the features selected in part 2 was very poor (˜0.5).

The results

We tried the following classification algorithms and evaluated their performance by the averaged accuracy of the 5-folds cross-validation results:

SVM
Logistic Regression
Boosted Trees
Neural Network
K Nearest Neighbors

The Logistic Regression performed the best, with accuracy of ˜0.64, thus we chose to use this classifier.